#2

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-149382
(43)Date of publication of application : 02.06.1998

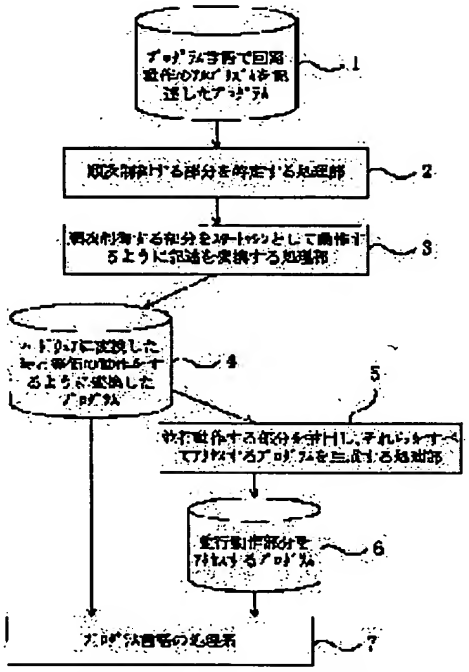
(51)Int.Cl. G06F 17/50
H01L 21/82

(21)Application number : 08-309049
(22)Date of filing : 20.11.1996

(71)Applicant : MATSUSHITA ELECTRIC IND CO LTD
(72)Inventor : MATSUKI TOSHIO

(54) DEVICE FOR DESIGNING ELECTRONIC CIRCUIT BY PROGRAM LANGUAGE

(57)Abstract:
PROBLEM TO BE SOLVED: To efficiently describe a circuit without complicating the description while executing circuit description by using general program language of which description property, debugging environment and execution environment are superior to that of hardware description language and to attain highly accurate simulation and debugging close to the operation of a real circuit.
SOLUTION: A specification processing part 2 specifies parts to be successively controlled from a program 1 in which circuit operation is described by the general program language. Then a conversion processing part 3 converts the description of the parts to be successively controlled by the general program language so as to drive it as a state machine and obtains a converted program 4. Then a program generation processing part 5 extracts parts to be driven in parallel from the converted program 4 and generates a program 6 for accessing all the extracted parts.



LEGAL STATUS

[Date of request for examination] 08.03.2001
[Date of sending the examiner's decision of rejection] 14.09.2004
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－149382

(43)公開日　平成10年(1998)6月2日

(51)Int.Cl.⁶	識別記号	ＦＩ	
Ｇ０６Ｆ　17/50		Ｇ０６Ｆ　15/60	６６４Ｋ
Ｈ０１Ｌ　21/82		Ｈ０１Ｌ　21/82	Ｃ

審査請求　未請求　請求項の数3　ＯＬ　（全８頁）

(21)出願番号　特願平8－309049

(22)出願日　平成８年(1996)11月20日

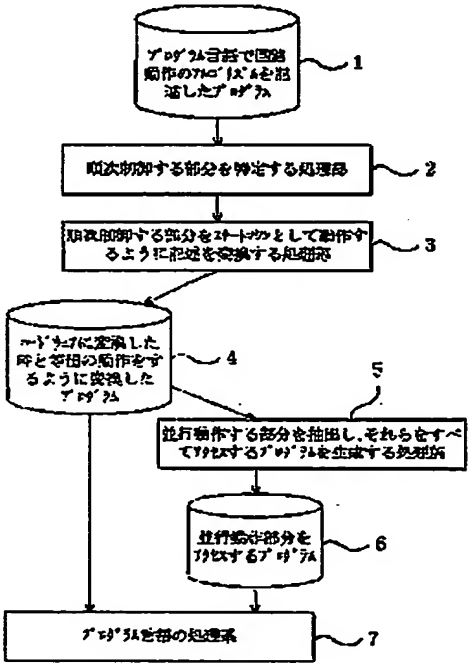
(71)出願人　000005821
松下電器産業株式会社
大阪府門真市大字門真1006番地

(72)発明者　松木　敏夫
大阪府門真市大字門真1006番地　松下電器産業株式会社内

(74)代理人　弁理士　前田　弘　（外２名）

(54)【発明の名称】　プログラム言語により電子回路を設計する装置

(57)【要約】
【課題】　ハードウエア記述言語よりも記述性、デバッグ環境、実行環境が優れている汎用プログラム言語を用いて回路記述を行いながらも、その記述を複雑化せず、効率良く記述できると共に、実際の回路の動作に近い精度の高いシミュレーション及びデバッグを行う。
【解決手段】　汎用プログラム言語で回路動作を記述したプログラム１の中から、順次制御する部分を特定処理部２で特定する。その後、変換処理部３で、前記順次制御する部分の記述を、ステートマシンとして動作するように汎用プログラム言語を用いて変換し、その変換後のプログラム4を得る。続いて、プログラム生成処理部５で、前記変換後のプログラム４の中から並行動作する部分を抽出し、この抽出部分の全てをアクセスするプログラム６を生成する。



MicroPatent® Worldwide PatSearch: Record 1 of 1

Family of JP10149382 How It Works

Family of JP10149382

No additional family members are found for this document

PatentWeb Home | Edit Search | Return to Patent List | Back to Record | Help

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開平10－149382
(43)公開日　平成10年(1998)６月２日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　17/50		Ｇ０６Ｆ　15/60	６６４Ｋ
Ｈ０１Ｌ　21/82		Ｈ０１Ｌ　21/82	Ｃ

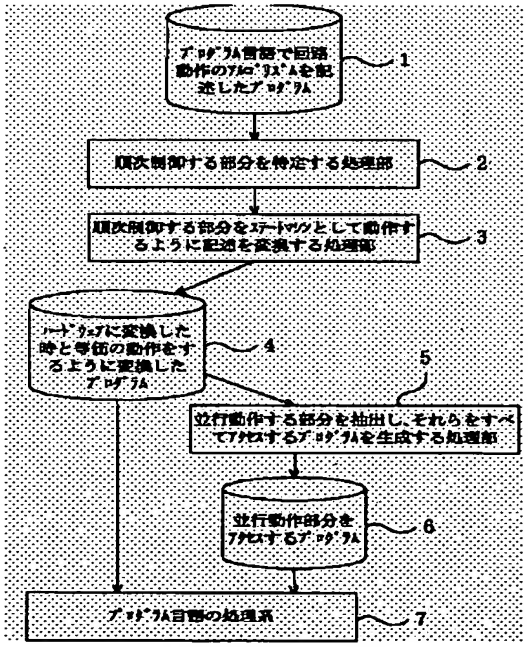
審査請求　未請求　請求項の数３　ＯＬ　（全８頁）

(21)出願番号　特願平8－309049
(22)出願日　平成８年(1996)11月20日

(71)出願人　000005821
松下電器産業株式会社
大阪府門真市大字門真1006番地
(72)発明者　松木　敏夫
大阪府門真市大字門真1006番地　松下電器産業株式会社内
(74)代理人　弁理士　前田　弘　（外２名）

(54)【発明の名称】　プログラム言語により電子回路を設計する装置

(57)【要約】
【課題】　ハードウエア記述言語よりも記述性、デバッグ環境、実行環境が優れている汎用プログラム言語を用いて回路記述を行いながらも、その記述を複雑化せず、効率良く記述できると共に、実際の回路の動作に近い精度の高いシミュレーション及びデバッグを行う。
【解決手段】　汎用プログラム言語で回路動作を記述したプログラム１の中から、順次制御する部分を特定処理部２で特定する。その後、変換処理部３で、前記順次制御する部分の記述を、ステートマシンとして動作するように汎用プログラム言語を用いて変換し、その変換後のプログラム４を得る。続いて、プログラム生成処理部５で、前記変換後のプログラム４の中から並行動作する部分を抽出し、この抽出部分の全てをアクセスするプログラム６を生成する。



【特許請求の範囲】
【請求項１】　汎用プログラム言語で回路動作を記述したプログラムから順次制御する部分を特定する特定処理部と、
前記特定処理部により特定された部分を汎用プログラム言語を用いてステートマシン化するための記述に変換する変換処理部と、
前記汎用プログラム言語で記述された回路を並行動作させるために、前記汎用プログラム言語で記述された回路の中の並行動作を行う関数を特定し、この特定した関数を並行動作させるためのプログラムを追加生成するプログラム生成処理部とを備えたことを特徴とするプログラム言語により電子回路を設計する装置。
【請求項２】　前記変換処理部により変換された変換プログラム及び前記プログラム生成処理部により生成された生成プログラムより成る変換生成プログラムで記述された回路を制御する制御プログラムを入力し、この制御プログラムの中から、ハードウエアへのアクセス部分を抽出する抽出処理部と、
前記抽出処理部により抽出されたアクセス部分の実行時刻を検出するプログラム、及び、このアクセス部分とその１つ前のアクセス部分との間の実行時間間隔を測定するプログラムとを、そのアクセス部分の直前に挿入する第１の追加処理部と、
前記第１の追加処理部により挿入したプログラムで得られる実行時間間隔に相当するクロック数分だけ前記変換生成プログラムを動作させるプログラムを、前記アクセス部分の直前に挿入する第２の追加処理部とを備えて、前記変換生成プログラムと前記制御プログラムとの同期を取ることを特徴とする請求項１記載のプログラム言語により電子回路を設計する装置。
【請求項３】　変換処理部により変換された変換プログラムで記述された回路の中でフリップフロップ又はラッチの生成が必要となるプログラム部分を探索し、フリップフロップ又はラッチを発生させる記述をハードウエア記述言語で前記変換プログラムに追加する追加処理部と、
前記変換プログラムで記述された回路の中で使用される変数を取得し、その変数の特性を決定し、その特性に応じてその変数をハードウエア記述言語でのモジュール入出力端子の記述に変換して前記変換プログラムに追加する第２の追加処理部と、
前記変換プログラム中の制御構文の表記をハードウエア記述言語記述の制御構文に変換する構文変換処理部とを備えることを特徴とする請求項１記載のプログラム言語により電子回路を設計する装置。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、Ｃ言語等の汎用のプログラム言語を用いて電子回路を設計する装置に関する。
【０００２】
【従来の技術】電子回路を設計するに際して、従来では、ハードウエア記述言語により回路記述を行い、高速なシミュレータで機能検証し、その後、論理合成ツールを用いて実際のハードウエアに直接対応するゲートレベルの回路を生成する手法が用いられてる。この手法は、最初からゲートレベルで開発するよりも効率良く大規模な回路を設計できる利点がある。
【０００３】最近では、ハードウエアシミュレータとプログラムデバッグツールとを組合わせて、マイクロプロセッサ上で動作するプログラムとハードウエアとを共にデバッグするツールが提供されている。
【０００４】
【発明が解決しようとする課題】しかしながら、ハードウエア記述言語では、ハードウエアをシミュレーション可能に時間を管理し、記述された処理の全てが時間の経過に従って並列に動作できるように考慮されているものの、次の欠点を有する。即ち、ハードウエア記述言語は、Ｃ言語等のプログラム言語と比較すると、記述性が劣る。また、ハードウエア記述言語で記述された回路の機能シミュレーションを行って回路のデバッグをする場合に、そのデバッグツールが、プログラム言語のデバッグツールほど洗練されていず、デバッグ方法で苦慮することが多い。更に、ハードウエア記述言語で書かれた回路をシミュレーションするシミュレータも高価であり、そのシミュレータの動作速度も遅いことが多い。更に、マイクロプロセッサ上で動作するハードウエア制御プログラムと、ハードウエアとを共にデバッグする場合であっても、ハードウエアの評価はハードウエアシミュレータに依存しているため、ハードウェアをハードウエア記述言語で記述する必要があり、従って、ハードウエアの記述性が良くなく、また、デバッグの難しさも依然残る。
【０００５】特開平４－２３０７６号公報では、高級言語で記述された回路を、マンマシンインターフエイスで対話的に操作を行って、ハードウエア記述言語に変換するものを提案するが、この提案の技術は、高級言語で書かれた回路記述の中で回路の動作とは無関係な部分を削除する機能と、逐次的な動作をする回路を生成するための技術であって、その適用範囲が限定される。
【０００６】一方、従来、Ｃ言語等のプログラム言語により回路の動作を記述し、シミュレーションを行って、回路の評価をする技術がある。この技術は、プログラム言語が、ハードウェア記述言語よりも記述性が良く、またデバッグ環境、実行環境等が既に整備されているので、デバッグを簡易に行うことができる。
【０００７】しかしながら、プログラム言語は、ハードウエア記述言語のように、並列に動作する回路を表現したり、クロック同期回路を表現しようとする場合には、

従来では、複雑な記述を行う必要があり、これ等の点においては、ハードウエア記述言語で記述した場合に比して、効率が悪くなる。仮に、複雑な記述をしなければ、プログラム言語が逐次処理を記述する言語であって、複数の処理を同時処理できない以上、実際の回路の動作とは全く異なった動きをして、精度の高いシミュレーションが行えない。

【０００８】本発明は、前記の諸点に着目してなされたものであり、その目的は、記述性が良くまたデバッグを簡易に行い得る汎用のプログラム言語を用いながら、その記述を複雑化せず、効率良く記述できると共に、実際の回路の動作に合致した精度の高いシミュレーションを行い得る設計装置を提供することにある。

【０００９】また、多くのシステムは、回路と、この回路を制御するプログラムとにより構成される。このシステムをシミュレーションにより検証する場合には、回路のみならず、制御プログラムをも回路と同時に共調してシミュレーションしなければならない。更に、シミュレーションで検証され且つ汎用のプログラム言語で記述された回路を正確に実際の回路に変換できなければ、シミュレーションで検証する意味が無くなる。

【００１０】そこで、本発明の更なる目的は、前記汎用のプログラム言語を用いながら実際の回路の動作に合致した精度の高いシミュレーションを行い得る回路記述のプログラムと、その記述された回路を制御する制御プログラムとの動作の同期を取ること、及び、そのような回路記述のプログラムを、その回路の論理合成が可能なハードウェア記述言語に変換することにある。

【００１１】

【課題を解決するための手段】前記の目的を達成するため、本発明では、プログラム言語の記述に、基本クロックに同期して動作する順次制御の概念と、複数の処理の同時並行処理の概念とを含むようにする。

【００１２】すなわち、請求項１記載の発明では、汎用プログラム言語で回路動作を記述したプログラムから順次制御する部分を特定する特定処理部と、前記特定処理部により特定された部分を汎用プログラム言語を用いてステートマシン化するための記述に変換する変換処理部と、前記汎用プログラム言語で記述された回路を並行動作させるために、前記汎用プログラム言語で記述された回路の中の並行動作を行う関数を特定し、この特定した関数を並行動作させるためのプログラムを追加生成するプログラム生成処理部とを備えたことを特徴とする。

【００１３】また、請求項２記載の発明は、前記請求項１記載の発明において、変換処理部により変換された変換プログラム及び前記プログラム生成処理部により生成された生成プログラムより成る変換生成プログラムで記述された回路を制御する制御プログラムを入力し、この制御プログラムの中から、ハードウエアへのアクセス部分を抽出する抽出処理部と、前記抽出処理部により抽出されたアクセス部分の実行時刻を検出するプログラム、及び、このアクセス部分とその１つ前のアクセス部分との間の実行時間間隔を測定するプログラムとを、そのアクセス部分の直前に挿入する第１の追加処理部と、前記第１の追加処理部により挿入したプログラムで得られる実行時間間隔に相当するクロック数分だけ前記変換生成プログラムを動作させるプログラムを、前記アクセス部分の直前に挿入する第２の追加処理部とを備えて、前記変換生成プログラムと前記制御プログラムとの同期を取ることを特徴とする。

【００１４】更に、請求項３記載の発明は、前記請求項１記載の発明において、変換処理部により変換された変換プログラムで記述された回路の中でフリップフロップ又はラッチの生成が必要となるプログラム部分を探索し、フリップフロップ又はラッチを発生させる記述をハードウエア記述言語で前記変換プログラムに追加する追加処理部と、前記変換プログラムで記述された回路の中で使用される変数を取得し、その変数の特性を決定し、その特性に応じてその変数をハードウエア記述言語でのモジュール入出力端子の記述に変換して前記変換プログラムに追加する第２の追加処理部と、前記変換プログラム中の制御構文の表記をハードウエア記述言語記述の制御構文に変換する構文変換処理部とを備えることを特徴とする。

【００１５】以上の構成により、請求項１記載の設計装置では、汎用のプログラム言語で記述された回路について、実際の回路に近い動作で精度の高いシミュレーションを行って、デバッグすることが可能である。また、実際のハードウエアへ変換することも可能である。

【００１６】また、請求項２記載の設計装置では、汎用プログラム言語で記述された回路のプログラムと、その回路を制御する制御プログラムとを、その両者で同期を取りながら、共に、シミュレーションの実行、デバッグ、及び評価を高い精度で行うことができる。

【００１７】更に、請求項３記載の設計装置では、汎用プログラム言語で記述された回路を、論理合成可能な汎用のハードウエア記述言語に変換して、ハードウエアを実現することを可能になる。

【００１８】

【発明の実施の形態】以下、本発明の実施の形態について図面を参照しながら説明する。

【００１９】図１は、請求項１記載の設計装置の実施の形態を示すブロック図である。

【００２０】同図において、１はプログラム言語を用いて回路動作のアルゴリズムを記述したプログラムである。このプログラムをＣ言語で記述した例を、図２、図４及び図６に掲げる。これ等の関数を適宜アクセスするプログラムを作成すれば、これ等の図の記述のままでもアルゴリズムの検証が可能である。これ等の図２、図４及び図６の記述を実行した場合には、ハードウエアの実

際の動作とは大きく異なることになるものの、システム全体の仕様の確認、アルゴリズムの確認、及びその記述された回路を制御する制御プログラムとの整合性の確認等、システム設計の初期に含まれる仕様レベルの不具合、アルゴリズムの不具合を極めて高速に検証することが可能である。回路部分のデバッグにも、プログラムのデバッグと同じツールが使えるので、プログラムのデバッグができる環境であれば、システム全体の評価が可能である。

【００２１】従来では、このレベルである程度の検証を行った後、回路図の入力装置を用いて回路を再度設計し、又はハードウエア記述言語に書き直していたため、ここで設計ミスが発生することが少なくない。また、この時点の記述では、既述の通り実際の回路動作と異なる部分が多くて、シミュレーションの精度が低く、一方、実際の回路と同じ動作をさせるような記述をしようとすれば、複雑な記述が必要となり、最初からハードウエア記述言語でシミュレーションする場合に比して、効率が悪くなる。

【００２２】前記図２、図４及び図６に掲げる記述には、基本クロックの変化に従って動作して行くハードウエアの動作概念が含まれず、また、この記述のままでは、ハードウエアの持つ同時並行動作もシミュレーションできない。一方、全てが同時並行動作をするのではなく、図２又は図４の関数の内部の処理のような場合には、逐次処理となり、ステートマシン動作が必要となる。

【００２３】そこで、次に、プログラム１を特定処理部２に入力し、この特定処理部２において、プログラム１の中から順次制御する部分を探し出し、特定し、その後、変換処理部３において、前記特定した順次制御する部分をステートマシンの動作を行う記述に変換し、その変換後のプログラム、即ちハードウエアに変換した時と等価の動作をするように変換したプログラム（変換プログラム）４を得る。図２の記述を変換した例を図３に、図４の記述を変換した例を図５に示す。

【００２４】続いて、前記変換処理部３で変換された変換プログラム４をプログラム生成処理部５に入力し、この処理部５において、前記プログラム４の中から、同時並行動作を行う部分を抽出し、この抽出部分の全てをアクセスするプログラム６を生成する。並行動作を行う部分は、予め、ルールを決定しておき、それをパターンマッチングにより探し出す。例えば、関数名にＳＵＢ＿やＬｉＢ＿が付いている場合には、ライブラリ的な回路として扱い、それ以外の関数は実体のある並行動作する回路であるとする。図７は、このルールを基に、図３、図５及び図６から同時並行動作の部分を抽出し、それ等を全てアクセスするように生成したプログラムの例である。

【００２５】７は、前記変換処理部３でステートマシンとして動作するよう変換された変換プログラム４及び、前記並行動作部分をアクセスするプログラム（生成プログラム）６を入力し、プログラム言語を処理するプログラム言語の処理系である。

【００２６】前記並行動作部分をアクセスするプログラム６をクロックの変化毎に実行しながら、前記変換プログラム４を実行すれば、実際のハードウエアの動作に極めて似たサイクルベースのシミュレーションが可能となる。例えば、図７のｒｕｎ＿ｈｄｃ（）を一度実行すると、クロック１つ分の実行となる。元のプログラム１と比較すると、シミュレーション速度は若干低下し、デバッグ性もやや劣化するものの、回路記述部分は実際のハードウエアの動作に近いシミュレーションが可能となるので、より一層に精度の高い評価が可能である。また、この段階でもプログラム言語のままであるので、回路記述のデバッグ環境は普通のプログラムのデバッグ環境と同じで足りる。

【００２７】図８は、請求項２記載の発明の実施の形態の設計装置のブロック図を示す。本実施の形態は、回路記述のプログラムと、その記述された回路を制御する制御プログラムとの同期を取るものである。

【００２８】同図において、８は、前記図１に示した変換プログラム４及び生成プログラム６で記述された回路を制御する制御プログラムである。この制御プログラム８には、必ず、図９に示すように、ハードウエア１４にアクセスする部分１５ａ，１５ｂ，１５ｃが存在する。前記アクセス部分１５ａでハードウエアにアクセスした後、その後のアクセス部分１５ｂに至るまでの時間は、特別な場合を除き、予め知ることは困難である。従って、その間、ハードウエアがどの程度のクロック変化分だけ動作したのか規定することは困難である。本実施の形態は、前記アクセス部分１５ａでのアクセス時刻と、その後のアクセス部分１５ｂでのアクセス時刻とを測定し、その時間間隔からハードウエアの動作時間を算出して、その時間分だけ変換プログラム４及び生成プログラム６（以下、変換生成プログラムという）を動作させて、その変換生成プログラムと制御プログラム８との同期を取ろうとするものである。

【００２９】即ち、図８において、制御プログラム８はハードウエアアクセス部抽出処理部９に入力され、この抽出処理部９は、ハードウエアにアクセスを行っている記述の位置を探索し、抽出する。その後、制御プログラム実行時間測定処理追加処理部（第１の追加処理部）１０は、前記抽出処理部９で探索、抽出されたハードウエアへのアクセス部分の直前に、その実行時刻を検出するプログラムを挿入すると共に、前回測定された前のアクセス部分の実行時刻と今回のアクセス部分の実行時刻との差分に基いて、ハードウエアへの前回のアクセスから今回のアクセスまでの時間を算出するプログラムを、ハードウエアへのアクセスを行う記述の直前に挿入する。

実行時刻の検出は、制御プログラム８を実行する環境が動作するオペレーティングシステムに対するシステムコール等で実現する。

【００３０】次いで、回路動作起動プログラム追加処理部（第２の追加処理部）１１は、前記第１の追加処理部１０で得られた実行時間から、この実行時間に相当するハードウエアのクロック数を算出し、そのクロック数分だけ回路記述のプログラム（即ち、変換プログラム４及び生成プログラム６）を動作させるプログラムを、制御プログラム８の中でハードウエアへのアクセスを行っている記述の直前に挿入する。本実施の形態では、図１に示した並行動作部分をアクセスするプログラム（生成プログラム）６をクロック数分だけ動作させるプログラムとなる。これにより得られた制御プログラム１２は、回路記述のプログラム６と同期したプログラムとなる。

【００３１】従って、この制御プログラム１２と回路記述の変換生成プログラムとを実行すれば、ソフトウエア及びハードウエアを含めたシステム全体の機能評価を精度良く行える。

【００３２】図１０は、請求項３記載の発明の設計装置の実施の形態のブロック図を示す。この実施の形態は、既述のように精度の高いシミュレーションを行い得るプログラム言語で記述された回路を、論理合成可能なハードウェア記述言語に変換するものである。

【００３３】即ち、図１０において、１６はハードウエアに変換した時と等価の動作をするように変換したプログラムであり、図１に示すステートマシンとして動作するように記述変更された変換プログラム４と同一である。

【００３４】追加処理部（第１の追加処理部）１７は、前記プログラム１６の中で、フリップフロップやラッチを発生させるプログラム記述を探索し、このフリップフロップ等を発生させる記述部分に、ハードウエア記述言語でフリップフロップやラッチが構築されるように記述を追加する。例えば、プログラムがＣ言語であれば、ｉｆ文を探索し、このｉｆ文の構造の解析を行い、このｉｆ文で記述された部分がセレクタ機能を表現していない場合、換言すれば、実際のハードウエアでフリップフロップ回路等を生成する必要がある場合には、ハードウエア記述言語でそのフリップフロップ回路等が生成される記述を追加する。例えば、ｉｆ文の中の実行文の部分でスタティック変数に値を設定している場合には、フリップフロップ回路が必要であるので、この部分には、Ｖｅｒｉｌｏｇ－ＨＤＬでは、ａｌｗａｙｓ文の記述を追加する。

【００３５】ハードウエア記述言語では、ハードウエアモジュールの入力端子と出力端子の定義を明確に行う必要があるのに対し、プログラム言語で書かれた回路記述には、入力端子や出力端子の概念がない。この点から、追加処理部（第２の追加処理部）１８は、プログラム１６の中で使用される変数を取得し、その変数の使用形態からその変数の特性を決定して、入力端子及び出力端子を判別し、それ等の端子をモジュールの記述の先頭部分に追加する。尚、ハードウエア記述言語でいうモジュールは、プログラム言語の関数に相当する。変数の特性の決定は、その変数がスタティック変数か、グローバル変数か、又はローカル変数か否か、及び、その変数へ値を代入しているか否かにより、決定する。

【００３６】更に、制御構文の表記の変換処理部（構文変換処理部）１９は、プログラム言語に特有の表現の制御構文を、ハードウエア記述言語の表現に変換する。この変換は、パターンマッチング法と置換により行う。これにより、ハードウエア記述言語による回路記述２０が生成される。この回路記述は、ハードウエア記述言語のシミュレータでもシミュレーションでき、更に、論理合成、レイアウトを経て、実際のハードウエアを提供することができる。

【００３７】

【発明の効果】以上説明したように、請求項１記載の発明によれば、安価で且つ数々の開発ツールが既に提供されている汎用プログラムの開発環境を使って、ハードウエアの仕様検討、アルゴリズムの検討、更には実際の回路動作に近い状態でのデバッグを、高速且つ快適に行うことが可能である。また、実際のハードウエアへ変換することが可能である。

【００３８】また、請求項２記載の発明によれば、汎用プログラム言語で記述された回路記述のプログラムと、その回路を制御する制御プログラムとの同期を取りながら、この両プログラムのシミュレーション実行、デバッグ、評価を高い精度で行うことができる。

【００３９】更に、請求項３記載の発明によれば、前記請求項１の発明により十分にデバッグされた，プログラム言語により書かれた回路記述を論理合成可能なハードウエア記述言語に変換して、最終目的であるハードウエアを実現することが可能である。しかも、ハードウエアの開発期間の多くを、高速に実行できるプログラム言語を使って進めることが可能となるので、従来よりも短い開発期間でハードウエアを開発できる。また、同時に上流工程においてソフトウエアの評価も可能であるので、システム全体の開発期間も大幅に短縮できる。更には、プログラム言語で書かれた回路記述からハードウエアの実現までを自動化することができ、設計ミスを減らすことができる。

【図面の簡単な説明】

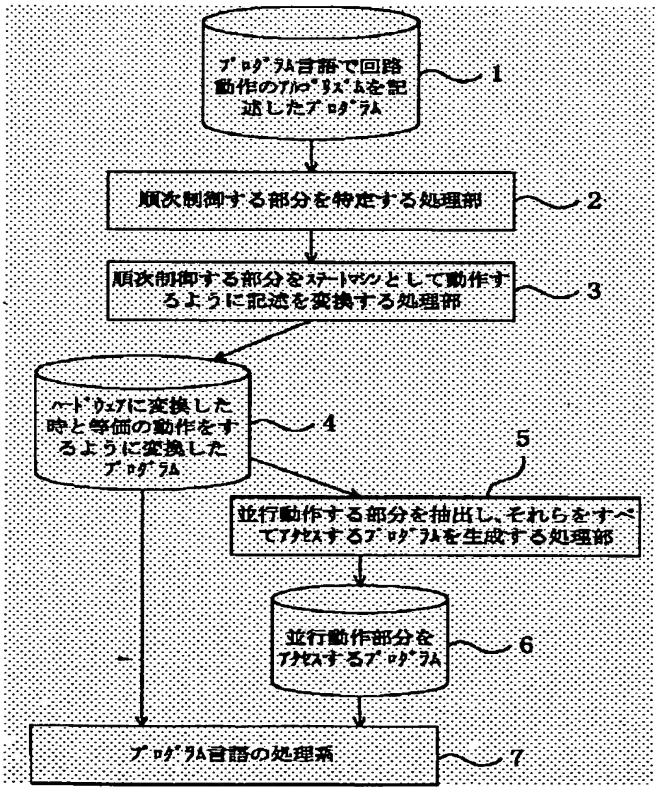
【図１】本発明の請求項１の設計装置の実施の形態を示すブロック図である。

【図２】プログラム言語で回路動作のアルゴリズムをＣ言語で記述した例を示す図である。

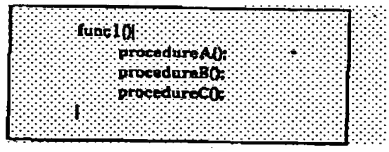
【図３】本発明の設計装置を用いて図２の記述を変換した例を示す図である。

【図４】Ｃ言語で回路動作のアルゴリズムを記述した例を示す図である。
【図５】本発明の設計装置を用いて図４の記述を変換した例を示す図である。
【図６】Ｃ言語で回路動作のアルゴリズムを記述した例を示す図である。
【図７】本発明の設計装置を用いて、図３、図５及び図６に例示した記述から同時並行動作部分をアクセスするプログラムの例を示す図である。
【図８】本発明の請求項２の設計装置の実施の形態を示すブロック図である。
【図９】ハードウェア（回路）と、この回路の制御プログラムとの間の制御関係を示す図である。
【図１０】本発明の請求項３の設計装置の実施の形態を示すブロック図である。
【符号の説明】
２　順次制御する部分を特定する処理部（特定処理部）
３　順次制御する部分をステートマシンとして動作するように記述を変換す処理部（変換処理部）
５　並行動作する部分を抽出し、それ等を全てアクセスするプログラムを生成する処理部（プログラム生成処理部）
７　プログラム言語の処理系
８　制御プログラム
９　ハードウエアアクセス部抽出処理部（抽出処理部）
１０　制御プログラム実行時間測定処理追加処理部（請求項２での第１の追加処理部）
１１　回路動作起動プログラム追加処理部（請求項２での第２の追加処理部）
１７　フリップフロップ又はラッチを発生するプログラム記述の探索及びそれ等を発生させる記述の追加処理部（請求項３での第１の追加処理部）
１８　プログラムの中で使われる変数の取得、変数の特性の決定、及び変数の記述の追加処理部（請求項３での第２の追加処理部）
１９　制御構文の表記の変換処理部（構文変換処理部）

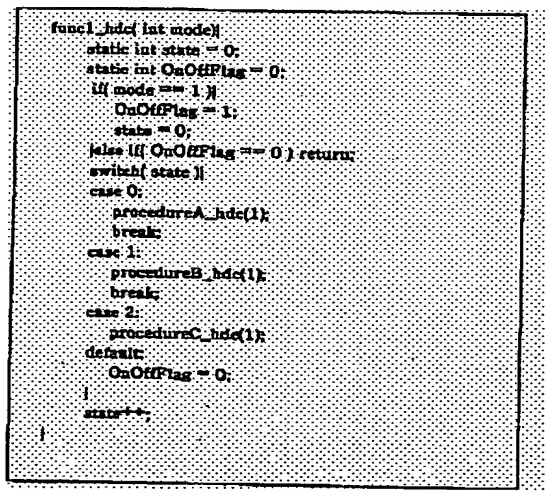
【図１】



【図２】



【図３】



【図４】

```
func2(){
    for( procedureA,conditionB,procedureC ){
        procedureD
    }
}
```

【図５】

```
func2_hdc(){
  static int state = 0;
  static int OnOffFlag = 0;
  if( mode == 1 ){
     OnOffFlag = 1;
     state = 0;
  }else if( OnOffFlag == 0 )   return;
  switch( state ){
  case 0:
     procedureA();
  default:
     procedureD();
  }
  procedureC();
  if( conditionB )   OnOffFlag = 0;
     state++;
}
```

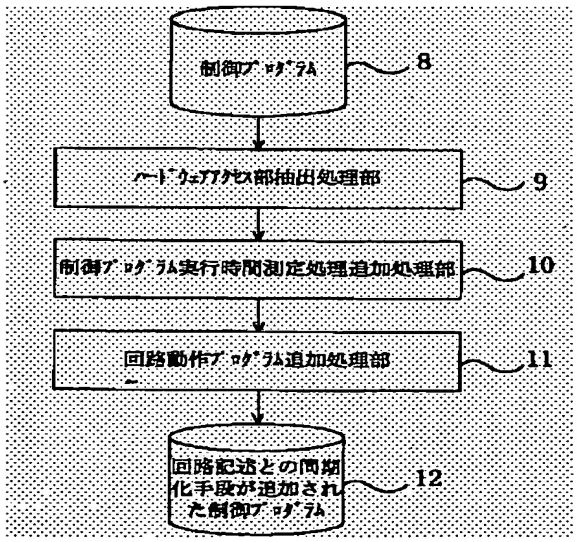
【図６】

```
setport(){
    if( wr && ( adr == 0x20 ) ){
        port0 = data;
    }
}

setmask(){
    if( wr && ( adr == 0x21 ) ){
        mask0 = data;
    }
}

outport(){
    outport0 = port0 & mask0;
}

selector(){
    if( outport0 == 0xab )
        bus0 = bus1;
    else
        bus0 = bus2;
}
```

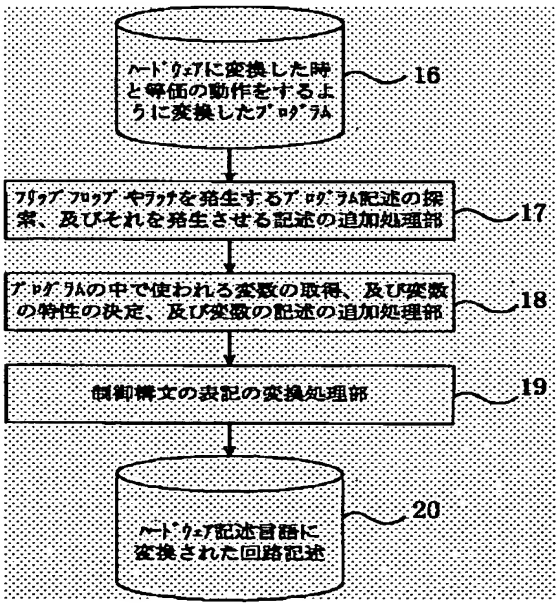
【図７】

```
run_hdc(){
    setport();
    setmask();
    outport();
    selector();
    func1_hdc();
    func2_hdc();
}
```

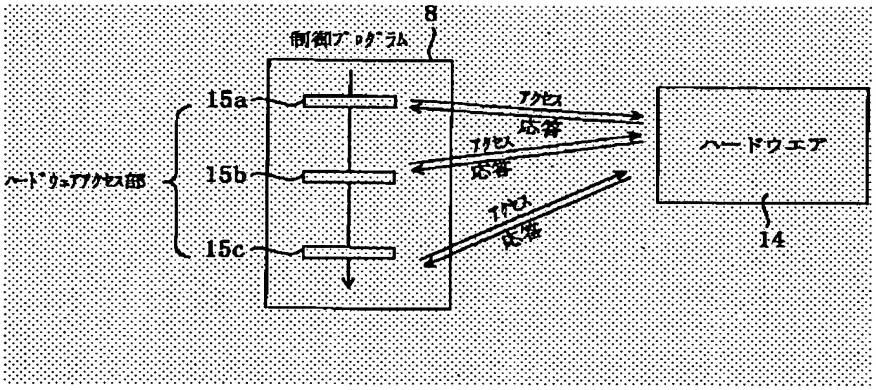
【図８】



【図１０】



【図９】


8
制御プログラム
15a
15b
15c
ハードウェアアクセス部
アクセス
応答
アクセス
応答
アクセス
応答
ハードウエア
14